continuity of data flow. Thus, in an effort to provide quality control for delivered content, attempts have been made to compare estimated data fetch times with estimated data send times to ensure that sufficient storage resources (*e.g.,* I/O operations per second) exist to fetch data in a period of time that is less than the period of time required to send the data to viewers in a timely manner. Such estimates may also be employed to ensure that sufficient storage processor memory (*e.g.,* buffer memory) exists to support the viewers. Such calculations have been employed to make decisions on whether or not new viewers may be admitted without adversely affecting delivery of content to existing viewers. In the past, such calculations have typically assumed conditions of static bit rate, constant data block size, and known number of viewers *i.e.,* by assuming a known number of streams and known service time required per stream.

Yet other attempts have been made to improve continuous content delivery by optimizing fetched block size. Techniques of this type include constant data length ("CDL") and constant time length ("CTL") methods. CDL methods fetch data blocks of constant size and have typically been employed for use in relatively homogenous content serving environments, or environments where demand for I/O operations is relatively constant, such as local area network ("LAN") environments. CTL methods fetch data using a fixed time interval rather than fixed data block size, so that fetched block size varies depending on instantaneous data transfer rate. CTL methods are typically employed for more heterogeneous content serving environments, or those environments where stream rates are variable, such as is often encountered in wide area network ("WAN") environments such as the Internet. However, it is often difficult to accurately estimate or measure data fetch times, especially for dynamically changing fetched block size schemes such as employed in CTL data fetching methods. Further, because CTL data fetching methods employ variable fetched block sizes it is also difficult to estimate memory requirements. Rapidly changing viewer identity and varying stream rates associated therewith further compromise the usefulness of such static-based calculations.

One example of past efforts at enhancing content delivery quality are methods directed towards admission control policies designed to support various scheduling algorithms. Examples of such admission control policies include a minimal buffer allocation algorithm used in a Continuous Media File System ("CMFS"), and a Quality Proportional Multi-subscriber

("QPMS") buffer allocation algorithm. Using the minimal buffer allocation algorithm, each data stream is assigned a minimal but sufficient buffer share for its read-ahead segment size in an attempt to ensure continuous playback. Using QPMS, the available buffer space is partitioned among existing data streams. However, both the minimal buffer allocation algorithm and the QPMS buffer allocation algorithm suffer from disadvantages. The minimal buffer allocation algorithm tends to generate an imbalance between storage load and memory consumption and requires re-calculation every time a new stream is introduced. The QPMS buffer allocation algorithm works to maximize memory consumption and also tends to generate an imbalance between memory and storage utilization. Thus, neither of these admission control policies perform well dynamically when various data streams are being added and removed.

SUMMARY OF THE INVENTION

Disclosed herein are methods and systems for I/O resource management that may be employed in an information delivery environment to manage I/O resources based on modeled and/or monitored I/O resource information, and that may be implemented in a manner that serves to optimize given information management system I/O resources, *e.g.*, file system I/O subsystem resources, storage system I/O resources, *etc.* The disclosed methods and systems may be advantageously implemented in the delivery of a variety of data object types including, but not limited to, over-size data objects such as continuous streaming media data files and very large non-continuous data files, and may be employed in such environments as streaming multimedia servers or web proxy caching for streaming multimedia files. Also disclosed are I/O resource management algorithms that are effective, high performance and which have low operational cost so that they may be implemented in a variety of information management system environments, including high-end streaming servers.

Using the disclosed algorithms, buffer, cache and free pool memory may be managed together in an integrated fashion and used more effectively to improve system throughput. The disclosed memory management algorithms may also be employed to offer better streaming cache performance in terms of total number of streams a system can support, improvement in streaming system throughput, and better streaming quality in terms of reducing or substantially eliminating hiccups encountered during active streaming.

Advantageously, the disclosed methods and systems may be implemented in an adaptive manner that is capable of optimizing information management system I/O performance by, for example, dynamically adjusting system I/O operational parameters to meet changing requirements or demands of a dynamic application or information management system I/O environment, such as may be encountered in the delivery of continuous content (*e.g.,* such as streaming video-on-demand or streaming Internet content), delivery of non-continuous content (*e.g.,* such as encountered in dynamic FTP environments), *etc.* This adaptive behavior may be exploited to provide better I/O throughput for an I/O subsystem by balancing resource utilization, and/or to provide better quality of service ("QoS") control for I/O subsystems. Thus, the disclosed methods and systems may be implemented to provide an application-aware I/O subsystem that is capable of high performance information delivery (*i.e.*, in terms of both quality and throughput), but that need not be tied to any specific application.

Further advantageously, the disclosed methods and systems may be deployed in scenarios (*e.g.,* streaming applications) that utilize non-mirrored disk configurations. When deployed in such non-mirrored environments, the disclosed methods and systems may be implemented to provide an understanding of the workload on each disk drive, and to leverage the knowledge of workload distribution in the I/O admission control algorithm.

In certain embodiments, the disclosed methods and systems may be employed so as to take advantage of relaxed or relieved QoS backend deadlines made possible when client side buffering technology is present in an information delivery environment. In certain other embodiments, the disclosed systems and methods may be additionally or alternatively employed in a manner that adapts to changing information management demands and/or that adapts to variable bit rate environments encountered, for example, in an information management system simultaneously handling or delivering content of different types (*e.g.,* relatively lower bit rate delivery employed for newscasts/talk shows, simultaneously with relatively higher bit rate delivery employed for high action theatrical movies). The capabilities of exploiting relaxed/relieved backend deadlines and/or adapting to changing conditions/requirements of an information delivery environment allows the disclosed methods and systems to be implemented